


Microsoft® Translator

English►Spanish x

Translate this page

Spanish ►

Microsoft® Translator



[Languages »](#) [C# »](#) [How To](#)

License: [The Code Project Open License \(CPOL\)](#)

Extract icons from EXE or DLL files

By [Tsuda Kageyu](#)

Extract all the variations of an icon, including the ones `ExtractIconEx()` can't extract.

C# (C# 3.0), Windows (Win2K, WinXP, Win2003, Vista), .NET (.NET 2.0), Dev

Posted: 9 Jun 2008

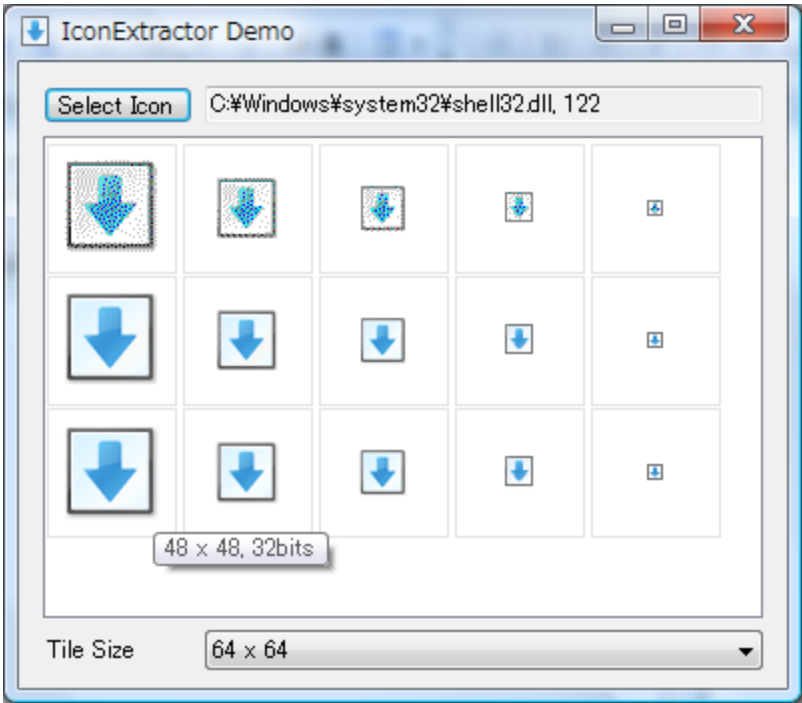
Updated: 2 Sep 2008

Views: 14,025

Bookmarked: 54 times



- [Download demo project - 13 KB](#)



Introduction

The class I introduce in this article is for extracting an icon from an EXE or DLL file. Though it's common to use the `ExtractIconEx()` Win32 API function for this purpose, `ExtractIconEx()` can't extract all the variations of an icon.

I had faced the problem, and made a class '`IconExtractor`' as the solution to the problem.

Using the code

To use this class, instantiate it first with the filename from which you want to extract the icons. You should dispose it after use.

Then, call the `GetIcon()` method with the index of the icon you want, to extract the specified icon. The index is same as the `nIconIndex` for `ExtractIconEx()`. This method returns a `System.Drawing.Icon` object containing all the variations in size and color depth. You can use it for `Form.Icon`, `NotifyIcon.Icon`, etc. The proper variation will be selected by the system.

Use the `SplitIcon()` static method if you want to split the variations into individual instances. This method returns an array of icons, each element containing one of the variations.

```
using TKageyu.Utils;

...

using (IconExtractor ie = new IconExtractor(@"D:\sample.exe"))
{
    Icon icon0 = ie.GetIcon(0);
    Icon icon1 = ie.GetIcon(1);

    ...

    Icon[] splitIcons = IconExtractor.SplitIcon(icon0);

    ...
}
```

How it works

The icons and other materials are embedded in the executable file in binary form. Those pieces of binary data are called 'resources'. They can't be read directly with .NET classes, because they are different from the managed resources that the .NET Framework can handle. They should be read with Win32 API functions such as `FindResource()`, `LoadResource()` etc.

This class reads binary data by using the functions in the `GetResourceData()` private method. The parameters given to the method are collected when the class is instantiated. The binary data is in similar form to an `.ico` file, and can be converted easily to be equivalent to an `.ico` file. It is given to the icon constructor after conversion, and will be parsed as read from an `.ico` file. The conversion is done in the `CreateIcon()` method.

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

About the Author


Tsuda Kageyu



Occupation: Software Developer

Location:  Japan

Member

Discussions and Feedback

 **7 messages** have been posted for this article. Visit <http://www.codeproject.com/KB/cs/IconExtractor.aspx> to post and view comments on this article, or click [here](#) to get a print view with messages.

Translate this page	<input type="text" value="Spanish"/>	
Microsoft® Translator		 

[PermaLink](#) | [Privacy](#) | [Terms of Use](#)

Last Updated: 2 Sep 2008

Editor: [Smitha Vijayan](#)

Copyright 2008 by Tsuda Kageyu
Everything else Copyright © [CodeProject](#), 1999-2009
Web15 | [Advertise on the Code Project](#)